## Procedures explanation:

The following SQL are designed to input the data to each tables. Each procedure serves a specific purpose and contributes to the management of properties, owners, tenants, leases, maintenance requests, vendors, financial transactions, and property listings.

1. **InsertRandomProperties**: This procedure inserts random property data into the "Properties" table. It generates random values for fields such as PropertyID, Address, PropertyType, Size, Amenities, Status, RentalPrice, and ListingStatus. This procedure helps populate the property database with diverse property information.
2. **InsertRandomOwners**: The purpose of this procedure is to insert random owner data into the "Owners" table. It generates random values for fields such as OwnerID, PropertyID, Contact Information, and Payment Details. Additionally, it assigns a random PropertyID from the "Properties" table to each owner. This procedure assists in establishing the relationship between properties and their respective owners.
3. **InsertRandomTenants**: This procedure inserts random tenant data into the "Tenants" table. It generates random values for fields such as TenantID, PropertyID, Lease Terms, Contact Information, and TenantStatus. Similar to the previous procedure, it assigns a random PropertyID from the "Properties" table to each tenant. This procedure helps create a diverse tenant base for the property management system.
4. **InsertRandomLeases**: The purpose of this procedure is to insert random lease data into the "Leases" table. It generates random values for fields such as LeaseID, PropertyID, TenantID, StartDate, EndDate, MonthlyRent, SecurityDeposit, and LeaseTerms. It also assigns random PropertyID and TenantID from the respective tables to each lease. This procedure enables the management of lease details and their association with properties and tenants.
5. **InsertRandomMaintenanceRequests**: This procedure, although commented out in the provided code, would typically insert random maintenance request data into the "MaintenanceRequests" table. Unfortunately, this part of the code is incomplete and has been commented out.
6. **InsertRandomVendors**: This procedure inserts random vendor data into the "Vendors" table. It generates random values for fields such as VendorID, Contact Information, Services Offered, HourlyRate, and Performance Ratings. This procedure aids in maintaining a database of vendors and their associated information.
7. **InsertRandomFinancialTransactions**: This procedure inserts random financial transaction data into the "FinancialTransactions" table. It generates random values for fields such as TransactionID, PropertyID, VendorID, TenantID, TransactionType, Amount, and TransactionDate. This procedure allows for the recording of financial transactions related to properties, vendors, and tenants.
8. **CreateListing**: This procedure creates a new listing by inserting data into the "Listings" table and updating the "Properties" table. It takes input parameters such as p\_PropertyID, p\_Description, p\_LeaseTerms, and p\_RentalPrice. This procedure enables the creation of property listings with relevant details, ensuring accurate representation in the property management system.
9. **UpdateListing**: This procedure updates an existing listing in the "Listings" table and updates the corresponding "Properties" table. It takes input parameters such as p\_ListingID, p\_Description, p\_LeaseTerms, and p\_RentalPrice. This procedure allows for modifications to listing information, ensuring up-to-date and accurate property listings.
10. **PublishListing**: This procedure publishes a listing by updating the "Listings" table and setting the PublishedDate. It also updates the "Properties" table to indicate that the property is listed. This procedure enables the activation and visibility of property listings within the property management system.
11. **UnpublishListing**: This procedure removes a listing by updating the "Properties" table and changing the ListingStatus to "Not Listed." This procedure facilitates the removal and deactivation of property listings within the property management system.
12. **SubmitMaintenanceRequest**: This procedure inserts a new maintenance request into the "MaintenanceRequests" table. It takes input parameters such as p\_PropertyID, p\_IssueReported, p\_UnitAffected, and p\_UrgencyLevel. This procedure allows tenants or property managers to submit maintenance requests for properties, ensuring efficient tracking and resolution of maintenance issues.
13. **AssignTaskToVendor**: This procedure assigns a maintenance request to a vendor by updating the VendorID, ScheduledDate, and RequestStatus in the "MaintenanceRequests" table. It takes input parameters such as p\_RequestID, p\_VendorID, and p\_ScheduledDate. This procedure facilitates the assignment of maintenance tasks to vendors, ensuring timely resolution of maintenance requests.
14. **UpdateMaintenanceRequestStatus**: This procedure updates the status of a maintenance request in the "MaintenanceRequests" table. It takes input parameters such as p\_RequestID and p\_RequestStatus. This procedure allows for the tracking and updating...of the status of maintenance requests, providing visibility into the progress and resolution of reported issues.

These SQL stored procedures collectively contribute to the smooth operation of a property management system. By generating random data and performing various actions such as creating properties, owners, tenants, leases, maintenance requests, vendors, financial transactions, and managing property listings, these procedures provide a fou-ndation for effective property management and administration.

In the provided code, several stored procedures are called to insert random data into the corresponding tables. For example, we insert 100 random records into the "Properties," "Owners," "Tenants," "Vendors," "FinancialTransactions," and "Leases" tables, respectively. The commented-out line suggests that there is an incomplete procedure for inserting random maintenance requests.

By executing these stored procedure calls, the property management system database is populated with a diverse set of random data, enabling testing and development of the system's functionality.